Maximo JSON API Overview

~~V1 - Updated: 11/25/2015~~
~~V2 - Updated: 12/08/2015~~
~~V3 - Updated: 12/17/2015~~
V4 - Updated: 10/23/2018

Contents

Introduction

In Maximo 7.6.0.2, a new JSON (Javascript Object Notation) API has evolved from the OSLC REST API (released 7.5.0.3), providing similar capabilities along with some Usability improvements. This API can operate on existing INTEGRATION (and OSLC) object structures and does not require the configuration of an OSLC Resource. The API does not support the use of OSLC namespaces and common properties, thus providing clean JSON data format. The API does support some of the OSLC standards around querying, such as the query parameters oslc.select and oslc.where.

This API follows Linked Data principles which implies that the API will use URI's to uniquely identify resources and would follow the HATEOAS (Hypermedia as the Engine of Application State). It is the recommended API for those needing to manage Maximo resources using JSON. In addition to MBOs, this API supports Attachments, Saved Queries and provides access to system level information.

Overview

The JSON API allows access to Maximo objects using an Integration object structure (Consumed by of INTEGRATION or OLSC) as the JSON Resource. The API supports the CRUD operations, as well as Query, using HTTP GET and POST.

Resources are defined as Integration object structures. A large number of these are provided out of the box with Maximo and can be used by the API. As well, new object structures that are configured are also usable with the API, providing their Consumed By value is either INTEGRATION or OSLC.

The majority of the examples used in this documentation will reference the out of box object structure, MXASSET, that supports integration of asset data that resides in Maximo.

Getting Started

If you are just starting out you should:

- Install a JSON Plug-in in your browser so that you can easily view the JSON data returned by the API
- In your Maximo environment, configure the host:port values in the System Property: mxe.oslc.webappurl (*reminder to do a Live Refresh)*

API Starting URL

For any Maximo environment, the Root, or Starting, URL is https://*host:port* /maximo/oslc/

User/Password

When working in a non-production environment, you have the option to pass the login id and password as query parameters to facilitate your development/testing.

?&_lid=wilson&_lpwd=wilson

Once set, they do not need to be provided on every request providing you keep your browser session open. If you have an active login to the Maximo UI in a separate tab of the browser, you will not have to provide login info for you API requests.

Alternatively you can use the maxauth HTTP header with the value as the base64 encoded user:password base64(wilson:wilson). There are online tools that will encode values.

lean=1

To avoid the use of OSLC namespaces, provide the lean query parameter (with a value of 1) on your initial request. If you maintain your browser session, you do not need to provide it on subsequent requests. If you forget to provide this value you will see the default namespace (spi) in your JSON data.

APIMETA

The Starting URL will return something similar to this:
https://*host:port* /maximo/oslc/?lean=1&_lid=wilson&_lpwd=wilson


```
{
   "systeminfo":
{
   "href": "http://host:port/maximo/oslc/systeminfo"
},
"apis": "http://host:port/maximo/oslc/apimeta",
"version":
{
   "href": "http://host:port/maximo/oslc/version"
},
"maxupg": "V7603-138",
"whoami":
{
   "href": "http://host:port/maximo/oslc/whoami"
},
"serverMembers":
{
   "href": "http://host:port/maximo/oslc/members"
},
"licenseKeys":
{
   "href": "http://host:port/maximo/oslc/license"
},
"currentDate": "2015-11-20T17:08:45-05:00",
"isPermanentLicense": true,
"language": "EN",
"href": "http://host:port/maximo/oslc",
"serviceProviders":
{
   "href": "http://host:port/maximo/oslc/sp"
},
"calendar": "gregorian",
"installedProducts":
   {
      "href": "http://host:port/maximo/oslc/products"
   }
}
```


The JSON data above provides some general system information and a number of links to other data/resources. The link for 'apis' is the one to use to start your access to data using the JSON api.

http://*host:port*/maximo/oslc/apimeta

This link will retrieve all the API metadata for all object structures that are Consumed By of INTEGRATION or OSLC. If you know the object

structure you are going to work with, you can retrieve the apimeta just for that (example using mxasset object structure):

http://*host:port*/maximo/oslc/apimeta/mxasset

APIMETA - MXASSET

Below is the metadata for the mxasset resource


```
{
   "schema": "http://host:port/maximo/oslc/jsonschemas/mxasset",
   "queryCapability":
[
          {
             "ispublic": true,
             "name": "All",
             "href": "http://host:port/maximo/oslc/os/mxasset"
          },
          {
             "ispublic": true,
             "name": "publicAssets",
             "javaMethod": true,
             "href": "http://host:port/maximo/oslc/os/mxasset?savedQuery=publicAssets"
          },
          {
             "title": "IT Stock in Stock Locations (non-Storeroom)",
             "ispublic": true,
             "name": "ITSTOCK",
             "href": "http://host:port/maximo/oslc/os/mxasset?savedQuery=ITSTOCK"
          },
             {
                "title": "X",
                "ispublic": true,
                "name": "LINKED-ASSETS",
                "href": "http://host:port/maximo/oslc/os/mxasset?savedQuery=LINKED-
          ASSETS"
             }
],
"authApp": "ASSET",
"title": "Asset Definition",
"useWith": "INTEGRATION",
"creationFactory":
[
      {
         "name": "default",
         "href": "http://host:port/maximo/oslc/os/mxasset"
      }
   ],
   "description": "",
   "href": "http://host:port/maximo/oslc/apimeta/mxasset",
   "osName": "MXASSET",
   "defaultPageSize": 100
}
```

The metadata for mxasset identifies a few key pieces of information. First is the Schema url which will provide a JSON schema for asset resource. Second is the queryCapability which identifies the urls to query the asset resource. Third is the creationFactory url which identifies the URL to create an asset resource.

JSON Schema

Below is a snippet of the JSON Schema for MXASSET where it lists all the fields of the resource with their length and types. The schema would reflect the configuration of objects and columns within the object structure.


```
{
    "title": "MXASSET",
    "$schema": "http://json-schema.org/draft-04/schema#",
    "description": "Json Schema for MXASSET",
    "properties":
{
    "description_longdescription":
{
    "title": "Details",
    "maxLength": 32000,
    "subType": "LONGALN",
    "usage":
    [
        "http://jazz.net/ns/ism/datatypes/smarter_physical_infrastructure#longaln"
    ],
    "type": "string"
},
"changedate":
{
```

```
        "title": "Changed Date",
        "maxLength": 10,
        "subType": "DATETIME",
        "type": "string"
    },
    "assetnum":
    {
        "title": "Asset",
        "maxLength": 12,
        "subType": "UPPER",
        "usage":
        [
            "http://jazz.net/ns/ism/datatypes/smarter_physical_infrastructure#upper"
        ],
        "type": "string"
    }
```


7.6.1 – JSON changes

Beginning in 7.6.1 support was added to process messages in a JSON format using a Publish/Invocation Channel and Enterprise Service. A flag was added Publish/Invocation Channel application to direct the framework to generate a JSON message from the object structure data, rather than XML. The MXXMLFILE End Point will support writing a file in JSON format, in addition to XML. The JSON file generated will be in a separate folder (jsonfiles) under the configured MIF Global Directory (mxe.int.globaldir).

An Enterprise Service can accept data in a JSON or XML format. When posting, if the Content-Type header has a value of application/json then processing will assume the payload is JSON, otherwise it will process as XML. File loading using the Data Import in the External Systems

application or using the XMLFILECONSUMER Cron Task have added a property to identify if the data being loaded is XML or JSON format.

Related documents will cover the details of the queryCapability and creationFactory.

Proceed to the documents that cover further details of the API such as querying and updating resources. Please send any corrections or suggestions to Tom Sarasin at tsarasin@us.ibm.com